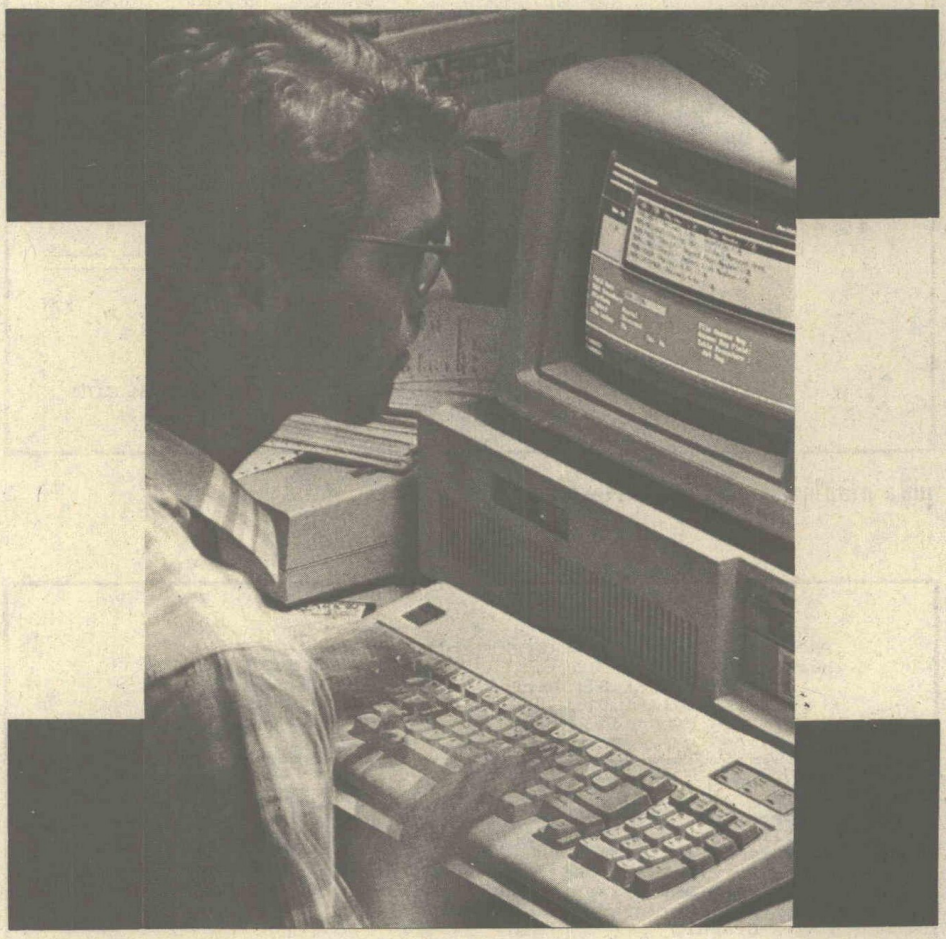
อนุชาติ ลาสนธิ

NP.COM
โปรแกรมห้ามกดคีย์

ห้ามกดคีย์มิใช่หมายความว่าดึงคีย์ตัวนั้นออกไม่ให้กด แต่ความหมายจริง ๆ ก็คือ กดคีย์นั้นได้ แต่ไม่มีผลใด ๆ ต่อโปรแกรมที่ใช้อยู่ขณะนั้น



โปรแกรม NP.ASM เขียนขึ้นโดยใช้ภาษาแอสเซมบลีของ 8088, 8086 หรือซีพียูตระกูลเดียวกันนี้ จุดประสงค์ที่เขียนโปรแกรมนี้ขึ้นมาเพื่อให้โปรแกรมทำหน้าที่เพียงอย่างเดียวคือ **ห้ามการกดคีย์** เอ๊ะ! ทำไมจึงต้องมีการห้ามการกดคีย์ด้วย ในเมื่อคีย์บอร์ดถูกนำมาไว้ให้สำหรับกด ใช่ครับ! คีย์บอร์ดถูกทำมาสำหรับกด แต่ถ้ากดคีย์ตัวไหนแล้วทำให้เกิดการสูญเสียก็ไม่ควรจะมีคีย์ตัวนั้นให้กด

สมมติต้องการจะพิมพ์งานซักอย่างโดยใช้โปรแกรมเวิร์ดโปรเซสเซอร์ วานให้เพื่อนช่วยพิมพ์ให้ แต่เพื่อนไม่ค่อยจะคุ้นเคยกับโปรแกรมตัวนี้นักบังเอิญไปกดถูกคีย์บางคีย์ที่ทำให้โปรแกรมเปลี่ยนไปทำงานในโหมดอื่น ถ้าในขณะนั้นเราไม่อยู่ เพื่อนอาจจะทำอยู่ 3 อย่างคือ 1. หยุดงานไว้ก่อนรอเรามา 2. ลองกดคีย์อื่น ๆ มั่วดู 3. เมื่อลองข้อ 2 ไม่ได้ก็จะนั่งรอ หากเพื่อนกล้าพอที่จะทำตามข้อ 2 ความเสียหายต่องานที่กำลังทำอยู่อาจจะเกิดขึ้นได้ แต่ถ้าเราเป็นคนรอบคอบซักหน่อยก็โดยทำการห้ามการกดคีย์ที่อาจจะทำให้เกิดความเสียหายไว้ก่อน ก็ให้ความปลอดภัยกับงานได้มากขึ้น และคนที่ใช้โปรแกรมก็ไม่ต้องกังวลกลัวว่าจะไปกดถูกคีย์ดังกล่าว คำว่า "ห้ามการกดคีย์" มิใช่หมายความว่า ดึงคีย์ตัวนั้นออกไม่ให้กดหรือถึงจะกดก็กดไม่ถูก แต่ความหมายจริง ๆ ก็คือ กดคีย์นั้นได้แต่ไม่มีผลใด ๆ ต่อโปรแกรมที่ใช้อยู่ขณะนั้น

การอินเตอร์รัพต์

การอินเตอร์รัพต์มีอยู่ 2 ประเภท คืออินเตอร์รัพต์ที่เกิดจากอุปกรณ์อินพุต เอาต์พุต (I/O) ส่งสัญญาณมาที่ขา INT ของซีพียู เรียกว่าฮาร์ดแวร์อินเตอร์รัพต์ เช่น คีย์บอร์ด (INT 09H), เวลาจริงของระบบ (INT 08H) เป็นต้น อันที่สองเกิดจากการใช้คำสั่ง INT เช่น INT 16H, INT 10H, INT 21H เป็นต้น

เรียกว่าซอฟต์แวร์อินเตอร์รัพต์ เมื่อเกิดอินเตอร์รัพต์ขึ้นไม่ว่าจะเป็นซอฟต์แวร์อินเตอร์รัพต์ หรือฮาร์ดแวร์อินเตอร์รัพต์ สิ่งที่ซีพียูจะต้องทำก็คือ หยุดงานที่กำลังทำขณะนั้นไว้ก่อนแล้วกระโดดไปทำงานที่โปรแกรมตำแหน่ง ๆ หนึ่ง เรียกโปรแกรมในตำแหน่งที่ซีพียูกระโดดไปทำงานเมื่อมีการอินเตอร์รัพต์นี้ว่า โปรแกรมตอบสนองการอินเตอร์รัพต์ (Interrupt service routine) ซอฟต์แวร์อินเตอร์รัพต์ยังแบ่งเป็น 2 อย่าง คือ อินเตอร์รัพต์ของไบออสหมายเลข 00-1FH และอินเตอร์รัพต์ของดอสหมายเลข 20-3FH จุดประสงค์ของแต่ละหมายเลขแสดงในรูปที่ 1 และ 2 ระบบอินเตอร์รัพต์ของไบออสนั้นจะเป็นการบริการขั้นพื้นฐานทั้งหมดให้กับอินเตอร์รัพต์ของดอส พูดง่าย ๆ ก็คือ อินเตอร์รัพต์ดอสจะทำอะไรจะต้องมาเรียกใช้อินเตอร์รัพต์ของไบออสอีกที

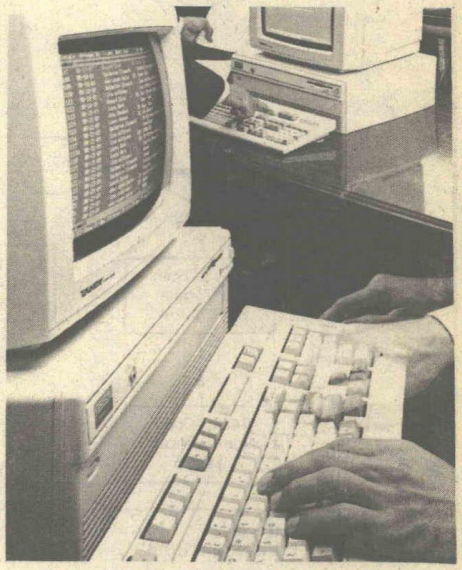
Address (Hex)	Interrupt Number	Name	BIOS Entry
0-3	0	Divide by Zero	D11
4-7	1	Single Step	D11
8-B	2	Nonmaskable	NMI_INT
C-F	3	Breakpoint	D11
10-13	4	Overflow	D11
14-17	5	Print Screen	PRINT_SCREEN
18-1B	6	Reserved	D11
1D-1F	7	Reserved	D11
20-23	8	Time of Day	TIMER_INT
24-27	9	Keyboard	KB_INT
28-2B	A	Reserved	D11
2C-2F	B	Communications	D11
30-33	C	Communications	D11
34-37	D	Disk	D11
38-3B	E	Diskette	DISK_INT
3C-3F	F	Printer	D11
40-43	10	Video	VIDEO_IO
44-47	11	Equipment Check	EQUIPMENT
48-4B	12	Memory	MEMORY_SIZE_DETERMINE
4C-4F	13	Diskette/Disk	DISKETTE_IO
50-53	14	Communication	RS232_IO
54-57	15	Cassette	CASSETTE_IO
58-5B	16	Keyboard	KEYBOARD_IO
5C-5F	17	Printer	PRINTER_IO
60-63	18	Resident BASIC	F600:0000
64-67	19	Bootstrap	BOOT_STRAP
68-6B	1A	Time of Day	TIME_OF_DAY
6C-6F	1B	Keyboard Break	DUMMY_RETURN
70-73	1C	Timer Tick	DUMMY_RETURN
74-77	1D	Video Initialization	VIDEO_PARMS
78-7B	1E	Diskette Parameters	DISK_BASE
7C-7F	1F	Video Graphics Chars	0

รูปที่ 1 หมายเลขอินเตอร์รัพต์ของไบออส

เกิดอะไรขึ้นบ้าง? เมื่อมีการกดคีย์

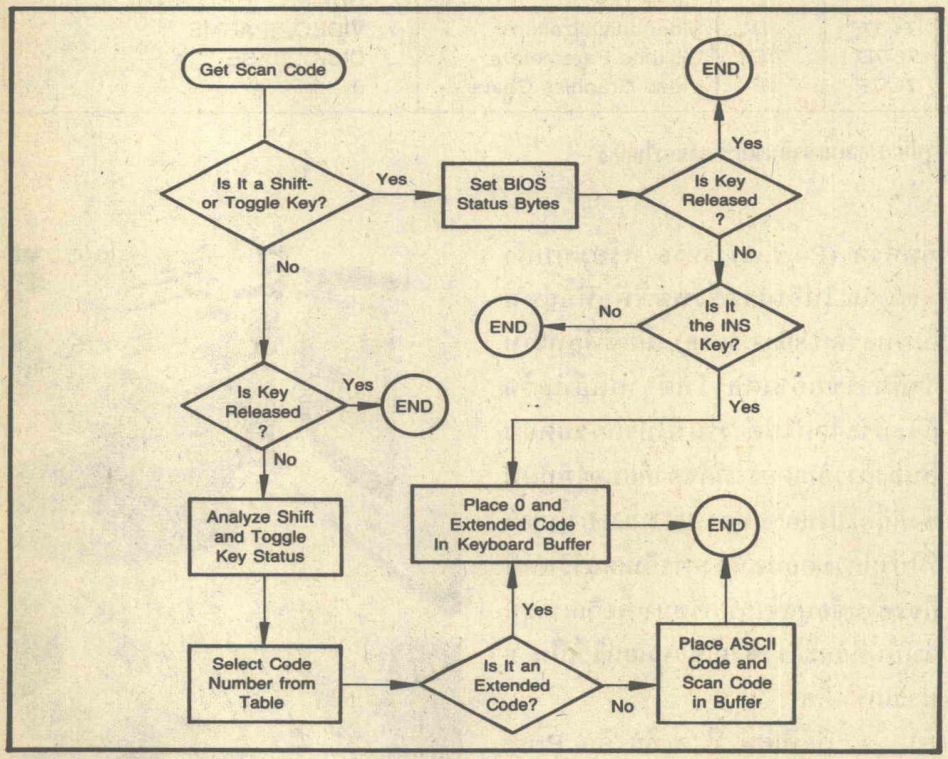
คีย์บอร์ดที่กดกันบ่อย ๆ คุณทราบหรือเปล่าว่า...เมื่อกดคีย์หนึ่งครั้ง เกิดอะไรขึ้นตั้งมากมายหลายขั้นตอน (ดังรูปที่ 3) ลองมาฟังดูพอคร่าว ๆ ในตัวคีย์บอร์ดจะมีไอซี Single Chip CPU เบอร์ 8048 หน้าที่ของซีพียูตัวนี้คือ คอยเฝ้าคีย์ทั้งหมดดูว่ามีคีย์ตัวไหนถูกกดบ้าง คีย์ตัวไหนถูกปล่อยแล้วบ้าง แล้วก็จะนำข้อมูลซึ่งเป็นค่าของตำแหน่งที่อยู่ของคีย์ตัวนั้น (เรียกว่าค่าสแกนโค้ด ดังรูปที่ 4 เป็นค่าสแกนโค้ดของคีย์แต่ละตำแหน่ง ซึ่งค่าเหล่านี้บริษัทผู้ผลิตได้กำหนดหรือออกแบบไว้แล้ว) ส่งไปไว้ที่พอร์ต (Port A) 8255 พร้อมกับส่งสัญญาณไปอินเตอร์รัพต์ซีพียูที่อยู่บนเมนบอร์ด (8088,8086) เมื่อซีพียูทราบว่ามีการขออินเตอร์รัพต์จากคีย์บอร์ดก็จะกระโดดไปทำงานที่โปรแกรมตอบสนองการอินเตอร์รัพต์ของคีย์บอร์ดทันที ซึ่งก็คือโปรแกรมของ INT 09H นั่นเอง ที่โปรแกรมตอบสนองการอินเตอร์รัพต์นี้ก็จะอ่านข้อมูลจากพอร์ตมาเพื่อวิเคราะห์ว่าเป็นคีย์อะไร ซึ่งผู้เขียนแยกไว้เป็น 5 ประเภท คือ

1. คีย์พิเศษ มี 2 คีย์ คือ Print screen, Break



Address (Hex)	Interrupt (Hex)	Function
80-83	20	DOS Program Terminate
84-87	21	DOS Function Call
88-8B	22	DOS Terminate Address
8C-8F	23	DOS Ctrl Break Exit Address
90-93	24	DOS Fatal Error Vector
94-97	25	DOS Absolute Disk Read
98-9B	26	DOS Absolute Disk Write
9C-9F	27	DOS Terminate, Fix In Storage
A0-FF	28-3F	Reserved for Dos
100-17F	40-5F	Reserved
180-19F	60-67	Reserved for User Software Interrupts
1A0-1FF	68-7F	Not Used
200-217	80-85	Reserved by BASIC
218-3C3	86-F0	Used by BASIC Interpreter while BASIC is running
3C4-3FF	F1-FF	Not Used

รูปที่ 2 หมายเลขอินเตอร์รัพต์ของดอส



รูปที่ 3 ขั้นตอนที่เกิดขึ้นเมื่อมีการกดคีย์

2. Toggle keys คือ Caps lock, Num lock, Scroll lock, Insert

3. Shift keys คือ Ctrl, Shift, Alt

4. ฟังก์ชันคีย์และคีย์ด่วน (hot keys) คือ F1-F10, Del, Home, End, Page up, Page Down และคีย์ควบคุมเคอร์เซอร์ทั้ง 4 คีย์

5. คีย์ตัวอักษรธรรมดา (Type-write keys) คือคีย์ของตัวอักษรธรรมดาที่มีรหัสแอสกีจาก 32 (space bar) -126 (~)

เมื่อซีพียูอ่านสแกนโค้ดจากพอร์ตมาแล้ว ก็จะมาวิเคราะห์ว่าเป็นคีย์ประเภทไหน ใน 5 ประเภทนี้ ถ้าเป็นประเภทที่หนึ่ง Print screen ก็จะทำให้เกิด (หรือเรียกใช้) INT 05H ขึ้นอีกตัวที่โปรแกรมตอบสนองการอินเตอร์รัพต์ของ INT 05H คือพิมพ์ข้อความที่จอภาพออกที่เครื่องพิมพ์ หรือถ้าเป็นคีย์ Break จะทำให้เกิด INT 1BH หยุดการทำงานไว้จนกระทั่งมีการกดคีย์หนึ่งคีย์ ก็คงจะทราบแล้วว่า INT 05H และ INT 1BH จริง ๆ นี้ก็เกิดจาก INT 09H นั่นเอง ถ้าเป็นคีย์ประเภทที่สอง หรือ Toggle key ในความหมายของ Toggle คือ ถ้าขณะนั้นติดเมื่อกดจะดับ และก็กลับกันถ้าดับกดแล้วจะติด คีย์ Toggle นี้จะรายงานสถานะการติดดับไว้ที่หน่วยความจำตำแหน่งที่ 0040:0017H ใน 4 บิตบน คีย์ประเภทที่สาม คือ Shift key ความหมายของ Shift key คือ เมื่อกดจะติดและปล่อยจะดับ ดังนั้นซีพียูจะทราบว่า Shift key จะติดก็ต่อเมื่อมีการกดคีย์แช่ไว้เท่านั้นแล้วซีพียูก็รายงานการกด Shift key ไว้หน่วยความจำตำแหน่งที่ 0040:0017H สี่บิตล่าง คีย์ประเภทที่สี่คือ ฟังก์ชันคีย์และ

Typewriter Keys

Key	Make Code	Key	Make Code	Key	Make Code
"1"	-2	"T"	-20	"L"	-38
"2"	-3	"Y"	-21	";"	-39
"3"	-4	"U"	-22	"'"	-40
"4"	-5	"I"	-23	"`"	-41
"5"	-6	"O"	-24	"\"	-43
"6"	-7	"P"	-25	"Z"	-44
"7"	-8	"["	-26	"X"	-45
"8"	-9	"]"	-27	"C"	-46
"9"	-10	"A"	-30	"V"	-47
"0"	-11	"S"	-31	"B"	-48
"-"	-12	"D"	-32	"N"	-49
"="	-13	"F"	-33	"M"	-50
"Q"	-16	"G"	-34	","	-51
"W"	-17	"H"	-35	"."	-52
"E"	-18	"J"	-36	"/"	-53
"R"	-19	"K"	-37	space bar	-57

Control Keys

Esc	-1	Ctrl	-29	Alt	-56
Backspace	-14	left shift	-42	CapsLock	-58
Tab	-15	right shift	-54	Numlock	-69
Enter	-28	PrtSc	-55	ScrollLock	-70

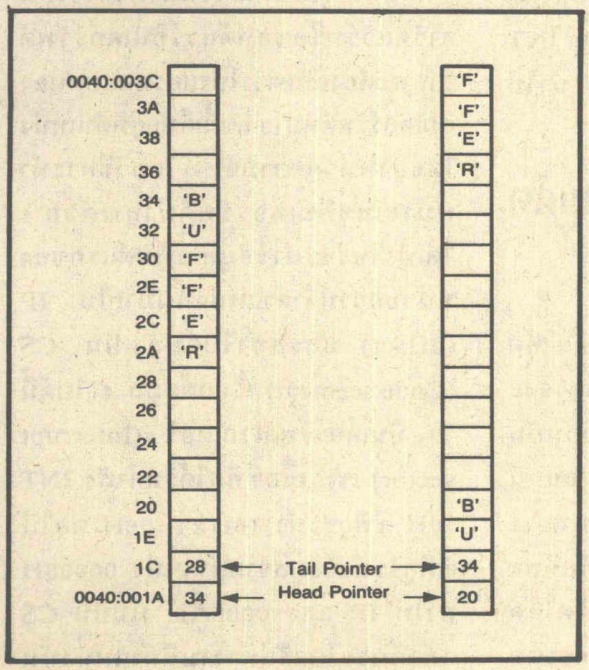
Function Keys

F1	-59	F5	-63	F9	-67
F2	-60	F6	-64	F10	-68
F3	-61	F7	-65		
F4	-62	F8	-66		

Keypad Keys

"7"	-71	"5"	-76	"3"	-81	
"8"	-72	"6"	-77	"0"	-82	
"9"	-73	"+"	-78	"."	-83	
"-"	-74	"1"	-79	Sys Req	-67	(AT only)
"4"	-75	"2"	-80	Phantom key	-55	(PC Jr only)

รูปที่ 4 ค่าสแกนโค้ดของคีย์ต่าง ๆ



รูปที่ 5 ลักษณะของคีย์บอร์ดบัฟเฟอร์

คีย์ด่วน เมื่อซีพียูตรวจสอบว่าเป็นคีย์ประเภทนี้จะนำสแกนโค้ดที่อ่านมาได้ไปเก็บไว้คีย์บอร์ดบัฟเฟอร์ (คีย์บอร์ดบัฟเฟอร์ คือหน่วยความจำตำแหน่งที่ 0040:001AH-0040:0030H มีลักษณะการเก็บเป็นแบบวนรอบ จึงต้องมีค่าสำหรับชี้ว่าคีย์ที่รับเข้ามาเริ่มเก็บที่ตำแหน่งไหน ถึงตำแหน่งไหน โดยค่าในหน่วยความจำที่ 0040:001AH เป็นตัวชี้หัวบัฟเฟอร์ (head pointer) ค่าที่ 0040:001CH เป็นตัวชี้ท้ายบัฟเฟอร์ (Tail pointer) ดังรูปที่ 5 พื้นที่สำหรับเก็บจริงมี 32 ไบต์ แต่ 1 คีย์ประกอบด้วย 2 ไบต์ ถ้าคีย์ที่กดเป็นคีย์ธรรมดาไบต์แรกจะมีค่าเป็นรหัสแอสกี ไบต์ที่สองเป็นค่าสแกนโค้ด ถ้าคีย์นั้นเป็นฟังก์ชันคีย์หรือ hot คีย์ ค่าของไบต์แรกจะเป็น 00 ไบต์ที่สองเป็นสแกนโค้ด ดังนั้นจริง ๆ แล้วบัฟเฟอร์จึงเก็บได้ทั้งหมด 16 คีย์) โดยทำให้ไบต์แรกมีค่าเป็น 00 ก่อน และถ้าเป็นคีย์ประเภทที่ 5 คือ คีย์ตัวอักษรธรรมดา ก่อนจะนำไปเก็บจะเซตให้ไบต์แรกเป็นรหัสแอสกีของคีย์ที่กด และไบต์ที่สองเป็นสแกนโค้ด เช่นกด 'A' สแกนโค้ดที่อ่านได้คือ 30 แต่ค่าที่นำไปเก็บในคีย์บอร์ดบัฟเฟอร์ ไบต์แรกคือ 65 และไบต์ที่สองคือ 30

INT 16H และ INT 09H ต่างกันตรงไหน

อินเตอร์รัพต์ที่เกี่ยวกับคีย์บอร์ดนั้นมี 2 อินเตอร์รัพต์ คือ INT 09H และ INT 16H อาจจะมีบางท่านแย้งผู้เขียนว่าแล้ว INT 21H Function 01H ละ ไม่เกี่ยวกับคีย์บอร์ดหรือเกี่ยวครับ! แต่ผู้เขียนหมายถึงอินเตอร์-

VALUE OF 2ND BYTE	CORRESPONDING KEYSTROKE
15	Shift + Tab ("back-tab")
16-25	Alt + Q to Alt + P (the top row of letters)
30-38	Alt + A to Alt + L (the middle row of letters)
44-50	Alt + Z to Alt + M (the bottom row of letters)
59-68	Function keys 1 to 10
71	Home
72	Cursor-up
73	PgUp
75	Cursor-left
77	Cursor-right
79	End
80	Cursor-down
81	PgDn
82	Ins
83	Del
84-93	Function keys 1 to 10 with the Shift key down
94-103	Function keys 1 to 10 with the Ctrl key down
104-113	Function keys 1 to 10 with the Alt key down
114	Ctrl + PrtSc
115	Ctrl + Cursor-left
116	Ctrl + Cursor-right
117	Ctrl + End
118	Ctrl + PgDn
119	Ctrl + Home
120-131	Alt + 1 to Alt + = (the top row of the keyboard)
132	Ctrl + PgUp

รูปที่ 6 เอกซ์เทนเด็ดโค้ดของคีย์ต่าง ๆ

รัพต์ของไบออส INT 21H นั้นเป็นอินเตอร์รัพต์ของดอสซึ่งก็ต้องเรียกใช้อินเตอร์รัพต์ของไบออสอีกที ผู้เขียนจึงถือว่ามีแค่ 2 ความแตกต่างของอินเตอร์รัพต์ทั้งสองตัวนี้เห็นได้ชัดมาก คือ INT 09H มีหน้าที่อ่านค่าสแกนโค้ดจากพอร์ตซึ่งส่งมาจาก 8048 มาวิเคราะห์ว่าสแกนโค้ดของคีย์ประเภทไหนแล้วก็จัดการตามคีย์ประเภทนั้น ๆ ส่วน INT 16H นั้นมีหน้าที่นำหรืออ่านข้อมูลที่ INT 09H เอามาเก็บไว้นำไปใช้งาน INT 16H จะมีฟังก์ชันใช้งาน 3 ฟังก์ชันหลัก ๆ (เครื่องบางรุ่นอาจจะมีมากกว่านั้น) คือฟังก์ชัน 0 นำข้อมูลจากคีย์บอร์ดบัฟเฟอร์ออกมา 1 ตัว ฟังก์ชัน 1 ตรวจสอบว่ามีข้อมูลในคีย์บอร์ดบัฟเฟอร์หรือไม่ และฟังก์ชันที่ 3 อ่านสถานะของ Shift key สรุปง่าย ๆ ก็คือ INT 09H นำมาเก็บไว้ INT 16H นำไปใช้งาน

สแกนโค้ดและเอกซ์เทนเด็ดโค้ด

สแกนโค้ดและเอกซ์เทนเด็ดโค้ดคงจะเคยได้ยินบ่อย ๆ บางท่านอาจจะคิดว่ามีความหมายหรือมีค่าเหมือนกันหรือบางท่านอาจจะยังไม่ทราบ สแกนโค้ด (scan code) คือข้อมูลขนาด 1 ไบต์ซึ่งเป็นค่าตำแหน่งของคีย์แต่ละคีย์ดังรูปที่ 1 ส่วนเอกซ์เทนเด็ดโค้ด (Extended code) นั้นคือข้อมูลขนาด 2 ไบต์ ซึ่งไบต์แรกจะต้องมีค่าเป็น 0 เสมอ เอกซ์เทนเด็ดโค้ดนี้เกิดจากโปรแกรมตอบสนองการอินเตอร์รัพต์ของ INT 09H เปลี่ยนจากสแกนโค้ดไปอีกที เช่น คีย์ F1 มีสแกนโค้ดเป็น 59 แต่มีเอกซ์เทนเด็ดโค้ดเป็น 00,59 ดูในรูปที่ 6

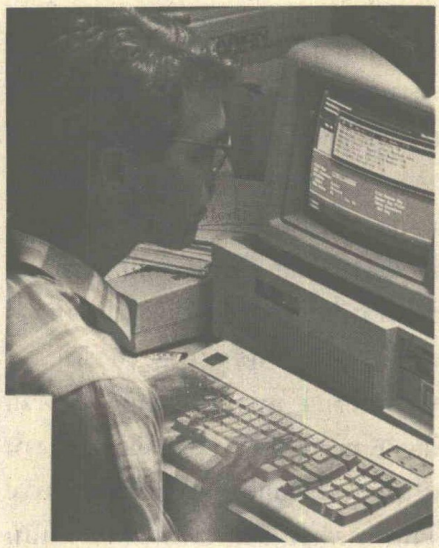
จะห้ามการกดคีย์ได้อย่างไร

คราวนี้มาดูหลักการว่าเราจะสามารถห้ามการกดคีย์ได้อย่างไร ดังที่กล่าวข้างต้นแล้วว่าเมื่อเกิดการอินเตอร์รัพต์ ซีพียูต้องกระโดดไปทำงานที่โปรแกรมตอบสนองการอินเตอร์รัพต์ของอินเตอร์รัพต์หมายเลขนั้นเสมอ ถ้าเป็นอินเตอร์รัพต์ของดอสโปรแกรมนี้จะอยู่ในแรม (RAM) หรือถ้าเป็นอินเตอร์รัพต์ของไบออสโปรแกรมส่วนนี้จะอยู่ในรอม (ROM) แล้วซีพียูทราบได้อย่างไรว่าโปรแกรมตอบสนองการอินเตอร์รัพต์แต่ละหมายเลขนั้นเก็บอยู่ที่ใด ถ้าจะให้เข้าใจดีต้องพูดถึงโครงสร้างของซีพียูเบอร์นี้ก่อน เมื่อซีพียูเกิดการอินเตอร์รัพต์ไม่ว่าจะเป็นฮาร์ดแวร์หรือซอฟต์แวร์อินเตอร์รัพต์ ซีพียูจะต้องค้นหาว่าโปรแกรมตอบสนองอินเตอร์รัพต์หมายเลขนี้เก็บอยู่ที่ตำแหน่งใดของหน่วยความจำ โดยวิธีการนำหมายเลขอินเตอร์รัพต์มาคูณด้วย 4 ได้เท่าไหร่จะนำข้อมูล 2 ไบต์แรกของหน่วยความจำตำแหน่งนั้นมาเป็น IP (offset) และสองไบต์หลังเป็น CS (Code segment) เรียกข้อมูล 4 ไบต์นี้ว่า อินเตอร์รัพต์เวกเตอร์ (Interrupt vector) เช่น เมื่อซีพียูได้รับคำสั่ง INT 16H ซีพียูจะนำ 16H×4=58H ต่อไป ซีพียูจะนำข้อมูลจากตำแหน่ง 00058H มาให้ IP และ 0005AH มาเป็น CS แล้วก็กระโดดไปทำงานที่ตำแหน่งนั้น

อินเตอร์รัพต์ของซีพียู 8088,8086 นั้นมีทั้งหมด 256 หมายเลข (00-FF) ในหนึ่งหมายเลขอินเตอร์รัพต์จะใช้ข้อมูลขนาด 4 ไบต์ (CS = 2, IP = 2) เป็นอินเตอร์รัพต์เวกเตอร์เพื่อบอกตำแหน่งของโปรแกรมตอบสนองการอินเตอร์รัพต์ ดังนั้นต้องใช้หน่วยความจำส่วนนี้เท่ากับ 256×4=1024 ตำแหน่ง หรือ 1 กิโลไบต์ เรียกพื้นที่ตำแหน่งนี้ว่าอินเตอร์รัพต์-เวกเตอร์เทเบิ้ล (Interrupt vector table) หลักการของการห้ามการกดคีย์ก็มีอยู่ว่าเมื่อรันโปรแกรมนั้นแล้วโปรแกรมจะต้องฝังตัว (Resident) ในหน่วยความจำของระบบ แต่ก่อนที่จะฝังตัวจะต้องเปลี่ยนอินเตอร์รัพต์เวกเตอร์ของ INT 09H หรือ INT 16H ตัวไหนก็ได้ ให้มาชี้ที่โปรแกรมซึ่งจะทำให้โปรแกรมที่ฝังตัวอยู่นั้นทำหน้าที่เป็นโปรแกรมตอบสนองการอินเตอร์รัพต์ของ INT 09H หรือ INT 16H ทันที ซึ่งก็หมายความว่าเมื่อมีการกดคีย์เข้ามาโปรแกรมที่ฝังตัวอยู่นี้จะรับรู้คีย์ที่กดก่อนโปรแกรมที่ใช้งานอยู่ขณะนั้น และถ้าโปรแกรมมีเพียงคำสั่ง IRET แสดงว่าตอนนี้คุณกดคีย์ตัวไหนเข้าก็ไม่มีผล หรือถ้าต้องการห้ามการกดเฉพาะบางคีย์ เช่น F1 โปรแกรมก็ต้องตรวจสอบว่าคีย์ที่กดเข้ามาเป็น F1 หรือไม่ ถ้าใช่ก็จะไม่สนใจคีย์นั้น แต่ถ้าไม่ใช่ก็ให้ทำงานเป็นปกติเหมือนโปรแกรมตอบสนองการอินเตอร์รัพต์อันเก่า

จะเปลี่ยนที่ INT 09H หรือ INT 16H ดี

หลักการก็ทราบแล้วว่า จะต้องเปลี่ยนอินเตอร์รัพต์เวกเตอร์ของคีย์บอร์ด คือ INT 09H หรือ INT 16H



ตัวใดตัวหนึ่งมาชี้ จะมาพิจารณาว่าจะใช้หมายเลขไหนดี เมื่อเลือก INT 09H แน่นอนที่สุดโปรแกรมที่ฝังตัวอยู่ต้องเป็นโปรแกรมตอบสนองฮาร์ดแวร์อินเตอร์รัพต์ โปรแกรมต้องเริ่มจากการอ่านสแกนโค้ดมาจากพอร์ต ตรวจสอบว่าเป็นสแกนโค้ดของคีย์ที่ห้ามการกดไว้หรือไม่ ถ้าใช่ก็ไม่ต้องนำค่านี้ไปเก็บในคีย์บอร์ดบัฟเฟอร์ และตอนจบของโปรแกรมก็ต้องจบแบบฮาร์ดแวร์อินเตอร์รัพต์ วิธีนี้จะทำให้การห้ามคีย์ได้ 100%

ที่เกริ่นมานี้ ถ้านำไปเขียนโปรแกรมจะยากและยาวพอสมควรเลยทีเดียว คนเขียนโปรแกรมต้องรู้เรื่องฮาร์ดแวร์บ้าง เช่น ต้องรู้ความหมายของค่าที่ส่งมาจาก 8048 หรือค่าที่อ่านมาจากพอร์ต ซึ่งไม่ใช่เป็นค่าสแกนโค้ดเสมอไป ส่วนการอ่านข้อมูลจากพอร์ตก็ต้องทราบว่าพอร์ต 8255 ต่อใช้งานในลักษณะใด จะมีวิธีการอ่านยังไง เนื่องจากพอร์ต 8255 บนเครื่องไอบีเอ็มพีซีนี้ 1 พอร์ต มีการต่อใช้งานหลายอย่าง เช่นที่พอร์ต A (60H) เป็นทั้งพอร์ตรับสแกนโค้ดจาก 8048 เป็นทั้งพอร์ตรายงานจำนวนดิสค์ไดรฟ์ ทั้งรายงานชนิดของจอภาพ และอีกอย่างคือการตรวจสอบ Shift คีย์ร่วมกับคีย์อื่น ๆ พร้อมทั้งการเขียนข้อมูลลงบัฟเฟอร์คีย์บอร์ด ซึ่งทั้งหมดนี้ถ้าคนเขียนโปรแกรมไม่คุ้นเคยกับภาษาแอสเซมบลีเท่าไหร่นัก ก็คงจะยากไม่น้อยเลยทีเดียว

แต่ถ้าใช้ INT 16H จะเขียนโปรแกรมง่ายขึ้นมาก ๆ เนื่องจากไม่เกี่ยวข้องกับฮาร์ดแวร์เหมือน INT 09H แต่การห้ามการกดคีย์จะไม่ 100% เนื่องจากว่าเมื่อผู้ใช้กดคีย์เข้ามาคีย์หนึ่งที่โปรแกรมตอบสนองอินเตอร์รัพต์ของ INT 16H ไม่สามารถห้ามได้ว่าไม่ให้เก็บค่านั้นไว้ในคีย์บอร์ดบัฟเฟอร์ เพราะว่าส่วนนี้ทำอยู่ที่โปรแกรมตอบสนองอินเตอร์รัพต์ของ INT 09H ดังนั้นหลักการห้ามการกดคีย์โดยใช้ INT 16H ทำได้เพียงตรวจสอบคีย์ที่กดเข้ามาถ้าเป็นคีย์ที่ห้ามการกดก็ให้เปลี่ยนเป็นค่าของคีย์อื่น ที่ไม่มีผลต่อโปรแกรมที่ใช้ในขณะนั้น อย่างเช่นโปรแกรมไซด์คิก คีย์ F1 คือการเรียก Help ถ้าต้องการห้ามคีย์ F1 โปรแกรมก็จะทำการตรวจสอบว่าคีย์ที่กดเป็นคีย์ F1 หรือไม่ ถ้าใช่ก็เปลี่ยนไปเป็นคีย์อื่นที่โปรแกรมไซด์คิกไม่สนใจหรือคีย์ที่ไซด์คิกไม่ใช้ เช่นคีย์ Alt +, Alt - เป็นต้น โปรแกรม NP.COM ก็ใช้หลักการนี้เช่นเดียวกัน คือเมื่อตรวจสอบว่าไม่ต้องการให้กดคีย์ไหนก็จะเปลี่ยนคีย์นั้นให้เป็น Alt- ซึ่งปกติแล้วโปรแกรมส่วนใหญ่จะไม่มีการใช้คีย์นี้อยู่แล้ว และนี่แหละคือเหตุผลที่ทำไมการใช้ INT 16H ห้ามการกดคีย์จึงไม่ได้ 100%

ใช้โปรแกรมอย่างไร

เมื่อได้ซอร์สไฟล์เรียบร้อยแล้ว นำไปผ่านการแอสเซมเบลอร์ จะใช้มาโครแอสเซมเบลอร์หรือเทอร์โบแอสเซมเบลอร์ก็ได้ แต่สุดท้ายก็จะได้ NP.

COM รูปที่ 7 คือตัวอย่างการแอสเซมเบลอร์โดยใช้มาโครฯ ถ้าซอร์สไฟล์ถูกต้องหมดจะเป็น 0 Error และ 0 Warning จะได้ NP.OBJ นำไปลิงค์ จะได้ NP.EXE ใช้โปรแกรม EXE2BIN เปลี่ยนจาก NP.EXE เป็น NP.COM พร้อมที่จะรันได้ ขนาดของโปรแกรมเมื่อเป็น .COM แล้วประมาณ 1 กิโลไบต์ ถ้าได้มากกว่า เช่นได้ 1028 ไบต์ ควรจะทำให้ได้ไม่เกิน 1 กิโลไบต์ โดยการไปลดขนาดของข้อความที่โชว์บรรทัดที่ 335-342 ให้เล็กลงกว่านั้น เพราะถ้าไฟล์ที่มีขนาดมากกว่า 1 กิโลไบต์แต่ไม่ถึง 2 กิโลไบต์ เช่น 1028 ไบต์ การเก็บลงดิสก์ต้องใช้เนื้อที่ถึง 2 กิโลไบต์

เมื่อติดตั้งหรือรัน NP.COM ครั้งแรก จะมีข้อความโชว์นิดหน่อย การทำงานก็ยังคงเป็นปกติต่อไป เมื่อต้องการห้ามการกดคีย์ตัวไหนให้ทำดังนี้ กด 'Alt +' เคอร์เซอร์จะกระโดดไปที่ตำแหน่ง 1,1 ตอนนี้คุณก็กดคีย์ที่ต้องการห้ามลงไปได้เลย ถ้าคีย์ที่กดเป็นตัวอักษรธรรมดาก็จะแสดงตัวอักษรนั้นด้วย แต่ถ้าเป็นฟังก์ชันคีย์จะแสดงตัวอักขระของแอสกี 1 แทน เมื่อพอแล้วจบด้วยคีย์ 'Enter' โปรแกรมจะกลับมาทำงานเป็นปกติเหมือนเดิม แต่คีย์ที่ทำการห้ามไว้ลองกดดูจะไม่มีผลถ้าต้องการให้เป็นปกติเหมือนเดิมคือไม่ต้องห้ามคีย์ใด ๆ ให้กดคีย์ 'Alt -' เช่นขณะนั้นใช้ไซด์คิกอยู่ต้องการห้ามคีย์ F2 และ 'Esc' ก็กด Alt + ตามด้วย F2, Esc แล้วกด Enter ทดลองกดสองคีย์ที่ห้ามไว้ดูจะไม่มีผลต่อโปรแกรมไซด์คิกเลย

```
A>MASM
  Microsoft (R) Macro Assembler Version 4.00 1985
  Copyright (C) Microsoft Corp 1981.1983,1984.All rights reserved.
  Source filename [.ASM]    : NP          [ENTER]
  Object filename [.OBJ]    :             [ENTER]
  Source listing  [NUL.LST] :             [ENTER]
  Cross-reference [NUL.CRF] :             [ENTER]
          48760 Bytes symbol space free
              0  Warning  Errors
              0  Severe   Errors
A>LINK NP;
  Microsoft (R) 8086 Object Linker Version 3.05
  Copyright (C) Microsoft Corp 1981.1983,1984.All rights reserved.
       Warning : no stack segment
A>EXE2BIN NP NP.COM
A>
```

รูปที่ 7 ขั้นตอนการแอสเซมเบลอร์โดยใช้มาโครแอสเซมเบลอร์

ปรับแต่งโปรแกรม

โปรแกรมนี้มีตัวแปรหลายตัว ที่สามารถเปลี่ยนแปลงให้เหมาะสมได้ดังนี้

- Kpop__up บรรทัดที่ 8 : เป็นเอกซ์เทนเด็ดโค้ดที่จะเรียกโปรแกรม NP.COM ให้ขึ้นมารับคีย์ที่ต้องการห้าม ปกติคือ 131 หรือเอกซ์เทนเด็ดโค้ดของคีย์ Alt +

- Kbd__non บรรทัดที่ 9 : เป็นเอกซ์เทนเด็ดโค้ดของคีย์ที่ต้องการยกเลิกการห้ามคีย์ทุกคีย์ ให้ทำงานเป็นปกติเหมือนเดิม ปกติคือ 130 หรือคีย์ Alt-

-Video__ram บรรทัดที่ 10 : เป็นหน่วยความจำของจอภาพที่ใช้ขณะนั้น ปกติคือ 0B000H ใช้สำหรับจอโมโนโครม ถ้าใช้จอสีเปลี่ยนเป็น 0B800H

- Kbd__buffer บรรทัดที่ 19 : เป็นจำนวนคีย์ที่ต้องการห้าม ปกติมีค่า 80 สามารถห้ามได้พร้อมกัน 40 คีย์ ถ้าต้องการมากกว่านั้นก็ให้เปลี่ยนได้เลย โดยต้องกำหนดค่าเป็น 2 เท่าของจำนวนคีย์เสมอ และได้มากที่สุด 80 คีย์

เจาะกึ๋นโปรแกรม

หลักการที่พูดมาทั้งหมดถ้าต้องการจะเข้าใจโปรแกรมมากกว่านั้น ก็ลองไล่การทำงานดูครับ

เมื่อโปรแกรมถูกรัน จะกระโดดไปทำงานที่ install (286) ทันที ส่วนนี้แหละที่จะทำหน้าที่ฝังโปรแกรมส่วนข้างบนไว้ อันดับแรกก็จะทำการอ่านค่าอินเตอร์รัพต์เวกเตอร์ของ INT 16H ขณะนั้นมาก่อน (290-292) แล้วทำการตรวจสอบว่าโปรแกรม NP.COM นี้เคยรันหรือยัง (295-297) โดยตรวจสอบค่าที่เคยฝังเอาไว้ (Mark) ถ้าเคยรัน

แล้วจะทำบรรทัดที่ (299-304) คือแสดงข้อความว่า 'NP.COM.....already แล้วก็ออกไปที่ดอส โดยไม่ฝังโปรแกรมไว้อีก เพราะจะทำให้เปลืองหน่วยความจำโดยเปล่าประโยชน์ แต่ถ้าเป็นการรันครั้งแรก จะกระโดดไปที่ No__ready (306) เก็บอินเตอร์รัพต์เวกเตอร์ของ INT 16H ขณะนั้นเอาไว้สำหรับเรียกใช้ทีหลัง (307-308) จากนั้นก็จะทำการเซตเคอร์เซอร์ไปที่ตำแหน่ง 1,1 พร้อมกับลบจอภาพ (310-320) พิมพ์ข้อความที่อยู่ในกรอบ (Show) ออกที่จอภาพ (322-324) ช่วงนี้สำคัญคือ (326-329) จะเปลี่ยนอินเตอร์รัพต์เวกเตอร์ของ INT 16H ขณะนั้นให้ไปชี้ที่ No__press (26 : ซึ่งก็คือจุดเริ่มต้นของโปรแกรมตอบสนองการอินเตอร์รัพต์อันใหม่นั่นเอง) และต่อไปเมื่อเปลี่ยนอินเตอร์รัพต์เวกเตอร์แล้วก็จะฝังส่วนนี้คือตั้งแต่บรรทัดที่ 18-284 อยู่ในหน่วยความจำตลอดเวลาเป็นอันว่าเสร็จสิ้นการเปลี่ยนอินเตอร์รัพต์เวกเตอร์พร้อมกับฝังโปรแกรมไว้ด้วย

คราวนี้เมื่อเกิด INT 16H ขึ้น ซีพียูจะไม่กระโดดไปทำโปรแกรมที่เคยไปแล้ว (เนื่องจากถูกเปลี่ยนอินเตอร์รัพต์เวกเตอร์มาชี้ที่โปรแกรม NP.COM แทน) แต่จะกระโดดไปทำงานที่บรรทัด 26 แทน เมื่อเข้ามาก็จะทำการตรวจสอบว่าเป็น INT 16H ฟังก์ชัน 0 หรือไม่ (เพราะว่าต้องการเฉพาะฟังก์ชัน 0 เท่านั้น เนื่องจากเป็นฟังก์ชันอ่านข้อมูลจากคีย์บอร์ดบัฟเฟอร์มาใช้งาน 30-31) ถ้าไม่ใช่ฟังก์ชัน 0 จะโอนการทำงานทั้งหมดให้โปรแกรมตอบสนองการอินเตอร์รัพต์อันเก่าของ INT 16H (33-34 คำสั่ง jmp old__vect ไม่ได้หมายความว่ากระโดดไปที่ old__vect การกระโดดแบบนี้เป็นการกระโดดแบบต่างเซกเมนต์ (Far) คือจะนำค่าในตัวแปร old__vect มาเป็น CS และ IP สำหรับที่จะกระโดดไป ซึ่งค่าที่เก็บใน old__vect ก็คือค่าอินเตอร์รัพต์เวกเตอร์อันเก่าของ INT 16H ที่เคยเก็บไว้นั่นเอง นั่นก็หมายความว่าโปรแกรม NP.COM เป็นโปรแกรมตอบสนองอินเตอร์รัพต์ของ INT 16H ฟังก์ชัน 0 เท่านั้น) แต่ถ้าใช่ฟังก์ชัน 0 ก็จะผ่านไปที่ pass__chk (36) ก่อนที่จะทำอะไรจะต้องเก็บรีจิสเตอร์ไว้ก่อน เพื่อป้องกันการเปลี่ยนแปลงสังเกตดูจะไม่มีการเก็บรีจิสเตอร์ AX ไว้เพราะว่าฟังก์ชัน 0 นี้ตอนออกไปจะต้องรายงานค่าที่อ่านมาจากคีย์บอร์ดบัฟเฟอร์ไว้ในรีจิสเตอร์ AX เมื่อเก็บรีจิสเตอร์เรียบร้อยแล้วจะทำการตรวจสอบว่าคีย์ที่กดเข้ามาใช่คีย์ Alt-หรือไม่ ถ้าใช่จะทำการยกเลิกการห้ามการกดคีย์ ถ้าไม่ใช่ Alt-ก็จะตรวจสอบต่อไปอีกว่าเป็น Alt+ ถ้าไม่ใช่จะผ่านไป ถ้าใช่จะเรียกใช้โปรแกรมย่อย Get__key (147) เพื่อรับคีย์ที่ต้องการห้ามการกด ส่วนบรรทัดที่ 79-110 จะทำการตรวจสอบดูว่าตอนนี้จะให้ห้ามการกดหรือไม่ ถ้าไม่ (Emty__check=0) จะออกจากโปรแกรม แต่ถ้าใช่ (Emty__check='!') ก็จะตรวจสอบดูว่าคีย์ที่กดเข้ามานั้นเป็นคีย์ที่ถูกห้ามไว้หรือไม่ แต่ถ้าไม่ใช่ก็ปล่อยไปเป็นปกติ แต่ถ้าใช่จะเปลี่ยนเอกซ์เทนเด็ดโค้ดของคีย์ตัวนั้นเป็น 130 (Alt-) แล้วก็ออกไปทำให้โปรแกรมที่ใช้อยู่ขณะนั้นได้รับคีย์นี้ด้วย ดังนั้นแสดงว่าคีย์ที่ถูกห้ามไว้ทุกคีย์จริง ๆ จะไม่ได้ถูกห้ามเพียงแต่ถูกเปลี่ยนให้เป็นคีย์ Alt- ซึ่งคีย์นี้เมื่อกดจะไม่มีผลใด ๆ ต่อโปรแกรมที่กำลังใช้ (เพราะว่าโปรแกรมส่วนใหญ่จะไม่ใช้คีย์นี้กัน) ส่วนโปรแกรมย่อยที่สำคัญก็มีดังนี้

- Clear__buf ล้างบัฟเฟอร์ที่ใช้เก็บคีย์ที่ต้องการห้าม เพื่อยกเลิกการห้าม
- Get__key รับคีย์ที่ต้องการห้ามไปเก็บไว้ใน Kbd__buffer
- Set__pos เซตเคอร์เซอร์ ไปที่ตำแหน่งแถวที่ DH คอลัมน์ที่ DL
- Get__pos อ่านตำแหน่งที่อยู่และขนาดของเคอร์เซอร์ขณะนั้น
- Abso เลื่อนเคอร์เซอร์ไป 1 คอลัมน์
- Print__char พิมพ์ค่าในรีจิสเตอร์ AL ออกจอภาพตำแหน่งเคอร์เซอร์
- Save__screen เก็บตัวอักษรและแอตทริบิวต์ของจอภาพในบรรทัดแรกไว้
- Rest__screen คืนตัวอักษรและแอตทริบิวต์ของจอภาพในบรรทัดแรก
- Save__cursor เก็บตำแหน่งและขนาดของเคอร์เซอร์ไว้
- Rest__cursor คืนตำแหน่งและขนาดของเคอร์เซอร์

ก่อนจากกัน

เมื่อต้องการติดตั้งโปรแกรม NP.COM พร้อมกับโปรแกรมประเภทฝังตัวอยู่ในหน่วยความจำอื่น ๆ เช่น ไทยไดรฟ์เวอร์หรือไซด์คิกต้องติดตั้งโปรแกรม NP.COM เป็นอันดับแรกก่อนโปรแกรมอื่น ๆ ไม่เช่นนั้นการห้ามคีย์จะไม่มีผลต่อโปรแกรมที่ฝังรวมอยู่ด้วย

รูปที่ 8 โปรแกรม NP.ASM

NP.COM โปรแกรมห้ามกดคีย์

```
                ;< Source file >
                ;
                ; *** NP.ASM ***
                ;
                ;-------------------------------------------------------------------------
                ;.........................................................................
Kpop_up         Equ  131                ;Scan code of Alt +
Kbd_non         Equ  130                ;Scan code of Alt
Video_ram       Equ  0B000H             ;Video buffer for monochrome

Cseg            Segment
                Assume Cs:Cseg,Ds:Cseg
                Org 100H
Start:
                jmp install
                ;-------------------------------------------------------------------------
Old_Vect        DD 00000000             ;Save interrupt vector
Kbd_Buffer      DB 80 dup(0)            ;Save scan code
Video_buffer    DB 160 dup(?)           ;Save screen
Position        DW 0000                 ;Save cursor position
Shape           DW 0000                 ;Save cursor shape
Emty_check      DB 00                   ;For check status of keyboard buffer
Mark            DW 0616H                ;For check already
                ;-------------------------------------------------------------------------
No_press        Proc Far
                Assume Cs:Cseg,Ds:Cseg

                sti                     ;reenable interrupts
                or ah,ah                ;check function 0
                jz pass_chk             ;

                Assume ds:nothing       ;if not function 0 then
                jmp old_vect            ;go back
                ;.........................................................................
Pass_chk:
                push bx                 ;
                push cx                 ;
                push dx                 ;
                                        ;
                push di                 ;
                push si                 ;
                                        ;
                push es                 ;
                push ds                 ;save altered registers

                push cs
                pop  ds
                ;.........................................................................
                Assume ds:nothing
                pushf                   ;
                call old_vect           ;get a character

                cmp ah,kbd_non          ;if scan code <> 130 (Alt -)
                jnz pass0               ;then jump to pass0
                mov emty_check,00       ;set key status is 00(no check)

                jmp exit
Pass0:
                cmp ah,kpop_up          ;If scan code <> 131 (Alt +)
                jnz pass1               ;then jump to pass1
                ;-------------------------------------------------------------------------
                push ax                 ;save scan code
                call save_screen        ;save screen page0 (1 row)
                call get_pos            ;get cursor position,shape
                call save_cursor        ;save cursor position,shape

                mov dh,0                ;
                mov dl,0                ;
                call set_pos            ;gotoxy(1,1)

                call clear_buf          ;clear keyboard buffer
                call get_key            ;get scan code for lock!
                call rest_screen        ;restore screen
                call rest_cursor        ;restore cursor position,shape

                jmp pass3
                ;-------------------------------------------------------------------------
Pass1:
                push ax                 ;save scan code
                cmp emty_check,00       ;if no check jump to pass3
                jz pass3

                mov si,offset kbd_buffer ;point to keyboard buffer
                mov cx,length kbd_buffer ;length of keyboard buffer
                shr cx,1                ;CX = CX/2
Kbd_chk:
                cmp al,0                ;if ascii code then
                jnz ascii               ;jump to ascii

                inc si                  ;else/ function key
                cmp ah,[si]             ;
                jnz no_func             ;check function key for lock!
                jmp pass2
Ascii:
                inc si                  ;
                cmp al,[si-1]           ;check ascii code for lock!
                jz pass2                ;
No_func:
                inc si                  ;
                loop kbd_chk            ;

                jmp pass3
Pass2:
                pop ax                  ;
                mov ah,kbd_non          ;make scan code to Alt -
                mov al,00               ;set status to no check
                jmp exit                ;
Pass3:
                pop ax                  ;restore scan code
                ;.........................................................................
Exit:
                pop  ds                 ;
                pop  es                 ;
                                        ;
                pop  si                 ;
                pop  di                 ;
                                        ;
                pop  dx                 ;
                pop  cx                 ;
                pop  bx                 ;restore registers
                ;.........................................................................
                Iret
No_press        Endp
                ;-------------------------------------------------------------------------
Set_up1         Proc  Near

                 mov si,offset kbd_buffer ;SI point to kbd_buffer
                 mov cx,length kbd_buffer ;CX = length of kbd_buufer
                 shr cx,1                ;CX = CX/2

                Ret
Set_up1         Endp
                ;-------------------------------------------------------------------------
Clear_buf       Proc  Near

                 call set_up1
                 mov al,0
Cls:
                 mov [si],al
```

```
141:                inc si
142:                loop cls
143:
144:               Ret
145: Clear_buf     Endp
146: ;-------------------------------------------------------------------------
147: Get_key       Proc  Near
148:
149:                call set_up1
150: Get1:
151:                mov ah,0                ;
152:                pushf                   ;
153:                call old_vect           ;get a scan code
154:
155:                cmp ah,28               ;if 'Enter' jump to ex_get
156:                jz ex_get               ;
157:
158:                mov [si],ax             ;
159:                inc si                  ;
160:                inc si                  ;
161:                call print_char         ;print a character
162:                loop get1               ;util cx = 0 or press 'Enter'
163: Ex_get:
164:                mov emty_check,'!'      ;set status to check
165:
166:               Ret
167: Get_Key       Endp
168: ;-------------------------------------------------------------------------
169: Set_pos       Proc  Near
170:
171:                mov bh,00               ;
172:                mov ah,02H              ;
173:                int 10H                 ;set cursor to x(dl),y(dh)
174:
175:               Ret
176: Set_pos       Endp
177: ;-------------------------------------------------------------------------
178:
179: Get_pos       Proc  Near
180:
181:                mov bh,00               ;
182:                mov ah,03H              ;
183:                int 10H                 ;get cursor position
184:
185:               Ret
186: Get_pos       Endp
187: ;-------------------------------------------------------------------------
188: Abso          Proc  Near
189:
190:                call get_pos
191:                inc dl
192:                call set_pos
193:
194:               Ret
195: Abso          Endp
196: ;-------------------------------------------------------------------------
197: Print_char    Proc  Near
198:
199:                push cx
200:
201:                cmp al,0                ;if function key
202:                jnz pc1                 ;then print chr(1)
203:                mov al,1
204: Pc1:
205:                mov cx,01               ;
206:                mov bh,00               ;
207:                mov bl,78H              ;
208:                mov ah,09H              ;
209:                int 10H                 ;print a character in A'
```

→

```
210:
211:                call abso
212:                pop cx
213:
214:               Ret
215: Print_char    Endp
216: ;-------------------------------------------------------------------------
217: Set_up2       Proc  Near
218:
219:                mov ax,video_ram        ;
220:                mov es,ax               ;ES point to video buffer
221:
222:                mov cx,80                ;character per line
223:                mov si,offset video_buffer  ;SI point to video buffer
224:                mov di,0000              ;DI point to top video buffer
225:
226:               Ret
227: Set_up2       Endp
228: ;-------------------------------------------------------------------------
229: Increment     Proc  Near
230:
231:                inc di
232:                inc di
233:                inc si
234:                inc si
235:
236:               Ret
237: Increment     Endp
238: ;-------------------------------------------------------------------------
239: Save_screen   Proc  Near
240:
241:                call set_up2
242: Loop_save:
243:                mov ax,es:[di]          ;get a character and attribute
244:                mov [si],ax             ;
245:                call increment          ;SI = SI+2,DI = DI+2
246:                loop loop_save          ;save chacters and attribute
247:
248:               Ret
249: Save_screen   Endp
250: ;-------------------------------------------------------------------------
251: Rest_screen   Proc  Near
252:
253:                call set_up2
254: loop_rest:
255:                mov ax,[si]             ;get a character and attribute
256:                mov es:[di],ax          ;
257:                call increment          ;SI = SI+2,DI = DI+2
258:                loop loop_rest          ;restore characters and attribute
259:
260:               Ret
261: Rest_screen   Endp
262: ;-------------------------------------------------------------------------
263: Save_cursor   Proc  Near
264:
265:                mov position,dx         ;save cursor position
266:                mov shape,cx            ;save cursor shape
267:
268:               Ret
269: Save_cursor   Endp
270: ;-------------------------------------------------------------------------
271: Rest_cursor   Proc Near
272:
273:                mov ah,02               ;
274:                mov bh,00               ;
275:                mov dx,position         ;
276:                int 10H                 ;restore cursor position
277:
278:                mov ah,01               ;
```

→

268 ไมโครคอมพิวเตอร์ 70

ซอฟต์แวร์

รูปที่ 8 (ต่อ) NP COM โปรแกรมห้ามกดคีย์

```
                mov bh,00                   ;
                mov cx,shap                 ;
                int 10H                     ;restore cursor shape

               Ret
Rest_cursor    Endp
;-----------------------------------------------------------------------
Install:
                push cs                     ;
                pop  ds                     ;DS = CS

                mov ah,35H                  ;function 35H(get the vector)
                mov al,16H                  ;
                int 21H                     ;get the vector INT 16H
                                            ;ES = CS,BX = offset

                mov ax,[bx-2]               ;AX = [BX-2]
                cmp ax,mark                 ;compare AX with mark
                jnz no_ready                ;if equal then jump to no ready
;.......................................................................
                mov ah,09H                  ;function 09H
                mov dx,offset ready         ;dx point to string(ready)
                int 21H                     ;Print '... NP.COM    alrady'

                mov ah,4CH                  ;functio 4CH
                int 21H                     ;exit to DOS
;.......................................................................
No_ready:
                mov word ptr cs:old_vect,bx  ;store offset of INT 16H to oldveq
                mov word ptr cs:old_vect+2,es  ;store CS of INT 16H to oldvect+
;.......................................................................
                mov dx,00                   ;
                call set_pos                ;gotoxy(1,1)

                mov ax,video_ram            ,
                mov es,ax                   ;ES point to video buffer

                mov di,0000                 ;DI = offset
                mov al,32                   ;32 = space
                mov ah,07                   ;Normal attribute
                mov cx,2000                 ;Character per page
                rep stosw                   ;Clear screen
;.......................................................................
                mov ah,09H                  ;function 09H
                mov dx,offset show          ;dx point to string(show)
                int 21H                     ;print string

                mov ah,25H                  ;function 25H
                mov al,16H                  ;
                mov dx,offset no_press      ;change the interrupt vector of INT
                int 21H                     ;point to No_press

                mov dx,offset install       ;set offset of resident routine
                int 27H                     ;quit,and routine stays resident
;-----------------------------------------------------------------------
;------------------------------ DATA AREA ------------------------------
Show:           DB '┌──────────────────────────────────────┐',10,13
                DB '│  *** NP.COM  ***                     │',10,13
                DB '│                                      │',10,13
                DB '│        for  IBM PC,XT,AT,386         │',10,13
                DB '│        --------------------          │',10,13
                DB '│         Com & Soft Group             │',10,13
                DB '│         01-26-91  12:30:00           │',10,13
                DB '└──────────────────────────────────────┘',10,13
```

```
                DB '$'

READY:          DB '......... NP.COM   already . ',7,10,13
                DB '$'
;-----------------------------------------------------------------------
CSEG            ENDS
                END START
;-----------------------------------------------------------------------

12:30:20  26/01/1991
```